Lecture 1: LLM and transformer foundations

Patrick Hall

Department of Decision Sciences

November 2, 2026

© 2026–2027 Patrick Hall (jphall@gwu.edu). Some rights reserved. Text and copy are licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0).

Agenda

Predicting the next word

Transformer Concepts

References

Tokenization turns text into supervised data

Vocabulary
$\mathcal{V} = \{$<pad>, see, spot, run, fast, <eot>$\}$, with IDs $0, \ldots, 5$.

The sentence see spot run fast <eot> becomes $[1, 2, 3, 4, 5]$. **Context-window length: $L = 4$ tokens.** Each prediction uses at most the four most recent token IDs; left-pad shorter prefixes with 0:

prefix	$\mathbf{x}^{(i)} \in \mathcal{X} \subset \{0, \ldots, 5\}^4$	$y^{(i)} \in \mathcal{Y} = \mathcal{V}$
see	[0, 0, 0, 1]	spot (2)
see spot	[0, 0, 1, 2]	run (3)
see spot run	[0, 1, 2, 3]	fast (4)
see spot run fast	[1, 2, 3, 4]	<eot> (5)

Supervised learning

The observed dataset is $\mathbf{X}, \mathbf{Y} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=0}^{N-1}$. Training produces a model g whose predictions $g(\mathbf{X}) = \hat{\mathbf{Y}}$ approximate $\mathbf{Y}$.

A one-hidden-layer MLP scores every possible next token

Each ID indexes a learned row of an embedding table $\mathbf{E} \in \mathbb{R}^{6\times 2}$.

For $\mathbf{x} = [x_1, \ldots, x_4]$:

$$\mathbf{E}_{\mathbf{x}} = [\mathbf{E}_{x_1}; \mathbf{E}_{x_2}; \mathbf{E}_{x_3}; \mathbf{E}_{x_4}] \in \mathbb{R}^{4\times 2}, \quad \mathbf{z} = \mathrm{vec}(\mathbf{E}_{\mathbf{x}}) \in \mathbb{R}^8.$$

$$\mathbf{h} = \tanh(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1), \quad \mathbf{s} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2, \quad \hat{\mathbf{p}} = \mathrm{softmax}(\mathbf{s}).$$

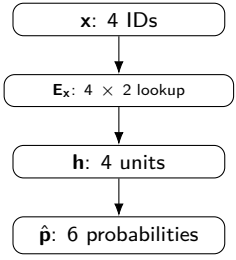
Here $\hat{p}_j = P_g(Y = j \mid \mathbf{x})$; categorical cross-entropy (multiclass log loss) for one example is $-\log \hat{p}_{y^{(i)}}$.

Gradient descent updates $\mathbf{E}, \mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2$.

Loss example: if $y^{(i)} = 4$ and $\hat{p}_4 = 0.80$, then $-\log(0.80) \approx 0.223$.

Layer flow: 4 token IDs $\to 4 \times 2 = 8$ embedding coordinates $\to$ 4 `tanh` units $\to$ 6 next-token probabilities.

Each layer's output matches the next layer's input: $(4, 2) \to 8 \to 4 \to 6$.

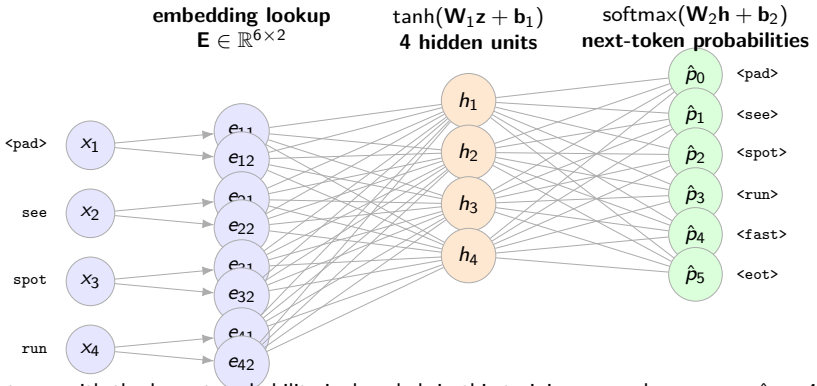


Why not predict ID 4 directly?

IDs are labels, not ordered quantities. Six softmax probabilities define a categorical prediction.

The example MLP

For $\mathbf{x} = [0, 1, 2, 3]$ (<pad>, see, spot, run), the context-window length is $L = 4$: each token contributes two learned coordinates. Every embedding coordinate connects to every hidden unit; every hidden unit scores every vocabulary item.



The output row with the largest probability is decoded; in this training example, $\arg\max_j \hat{p}_j = 4$, or fast.

Detokenization makes the prediction readable

For the context `see spot run`, tokenization gives $\mathbf{x} = [0, 1, 2, 3]$. After the MLP:

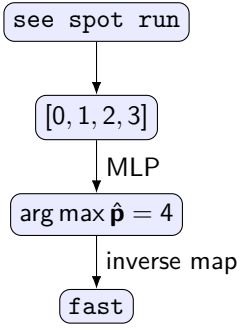
$$\hat{\mathbf{p}} = \left[\hat{p}_{\texttt{<pad>}}, \hat{p}_{\texttt{see}}, \hat{p}_{\texttt{spot}}, \hat{p}_{\texttt{run}}, \hat{p}_{\texttt{fast}}, \hat{p}_{\texttt{<eot>}}\right].$$

Decode

$\hat{y} = \arg\max_{j \in \{0,\ldots,5\}} \hat{p}_j$ is an ID. The inverse vocabulary map converts it back to text: $\hat{y} = 4 \mapsto$ `fast`.

Generate another token

Append the decoded ID, shift the context to $[1, 2, 3, 4]$, and predict again. A learned model should then assign high probability to <eot>.



What the two-dimensional embedding plot shows

The notebook plots the learned 2-D vector $\mathbf{E}_j \in \mathbb{R}^2$ associated with each vocabulary token after training.

- Tokens used in similar predictive contexts can become nearby; this is an empirical pattern, not a guarantee in a tiny dataset.
- The axes themselves have no fixed linguistic interpretation: rotating or reflecting all embeddings can leave equivalent downstream behavior after compensating weights.
- The plot is useful for inspecting the representation, while softmax probabilities remain the model's next-word output.

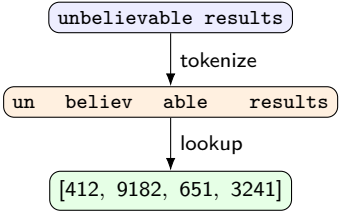
Notebook correspondence

Same vocabulary, four padded examples, 2-D token embeddings, 4 hidden units, cross-entropy training, decoding, and an embedding scatter plot.

$$\text{text} \rightarrow \text{IDs} \rightarrow \text{embeddings} \rightarrow \text{probabilities} \rightarrow \text{text}$$

Tokens are the model's units of text

- A tokenizer maps text to IDs from a fixed vocabulary; detokenization reverses the map.
- Subword vocabularies reuse frequent pieces, so rare words need not become unknown words.
- Context limits and API usage are measured in tokens, not words or characters.



Important consequence

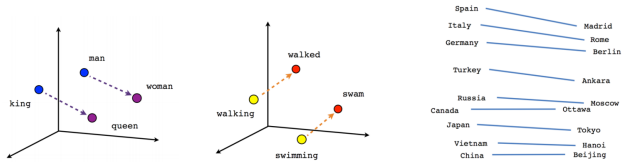
The same visible word can use different token pieces across tokenizers; token counts therefore depend on the model's vocabulary.

Source: Sennrich et al. (2016), subword units.

Embeddings turn discrete IDs into learned vectors

An embedding table maps token ID j to a dense vector $\mathbf{E}_j \in \mathbb{R}^d$.

- Vectors are learned from predictive objectives.
- Relative geometry can encode useful regularities.
- In a transformer, position information is added or otherwise incorporated before attention.



Illustrative embedding relations: similar transformations appear as related vector offsets.

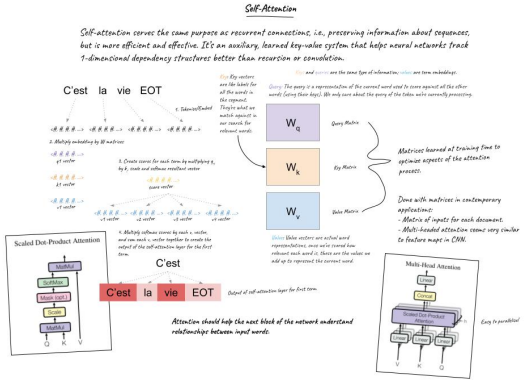
Source: Mikolov et al. (2013), distributed word representations.

Attention mixes information across token positions

At each token position, learned linear transformations map the input representation to a query, key, and value vector:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V.$$

- For each query, softmax weights determine how value vectors are mixed into the output.
- Multiple heads can learn different relationships.
- In self-attention, queries, keys, and values are derived from the same sequence of tokens.

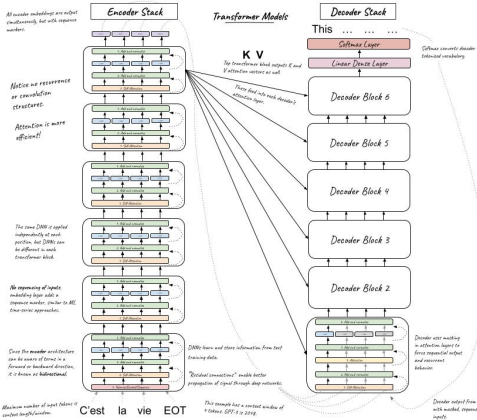


Queries score keys; attention weights mix value vectors across token positions.

Source: Vaswani et al. (2017), scaled dot-product attention.

Transformers stack attention and feed-forward blocks

- Residual connections and normalization stabilize deep stacks.
- Encoder stacks build bidirectional representations.
- Decoder stacks use causal masking: position t cannot inspect future tokens.
- Many contemporary LLMs are decoder-only transformers.



An encoder–decoder transformer stacks attention and feed-forward blocks.

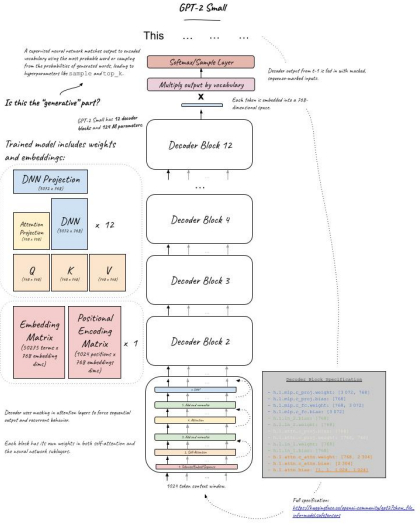
Source: Vaswani et al. (2017), transformer architecture.

Generation

At step t, the model produces logits $\mathbf{s}_t$ and a categorical distribution:

$$p(x_{t+1} \mid x_{\leq t}) = \mathrm{softmax}(\mathbf{s}_t / T).$$

- Greedy decoding chooses the highest-probability token.
- Lower T sharpens the probability distribution; higher T flattens it. Top-k and top-p truncate the candidate set.
- The chosen ID is appended and the forward pass repeats until a stop condition.



Visual: A GPT-2-style decoder maps contextual token representations to next-token probabilities.

Source: Holtzman et al. (2020), decoding and nucleus sampling.

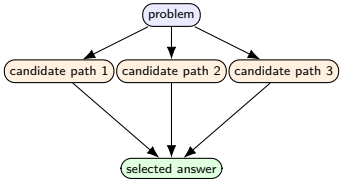
Inference-time scaling trades resources for solution quality

Instead of changing model weights, an inference procedure can spend more compute by:

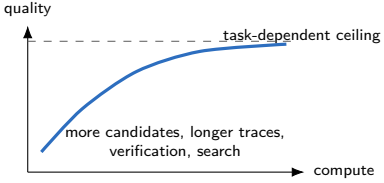
- generating longer deliberations;
- sampling several candidate solutions;
- verifying, ranking, or revising candidates;
- searching a tree of possible continuations.

Returns can plateau or reverse: the best allocation depends on task difficulty and the verifier.

Multiple candidates and selection



Compute–quality tradeoff

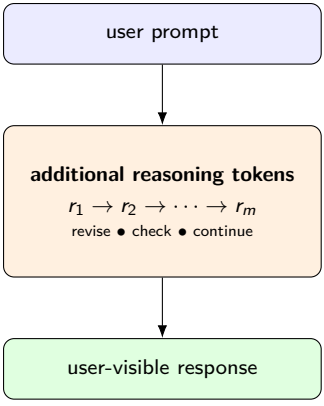


Top: several candidate reasoning paths feed a selected answer. Bottom: additional inference compute can improve solution quality, with diminishing returns.

Source: Snell et al. (2024), test-time compute; Promontory AI (2026), reasoning effort and cost.

Reasoning methods allocate intermediate computation

- Chain-of-thought prompting uses intermediate text to improve performance on some multi-step tasks.
- Reasoning models use training and decoding procedures that spend additional computation before returning a final answer.



A reasoning model generates additional reasoning tokens before issuing a user-visible response.

A trace is not a causal explanation

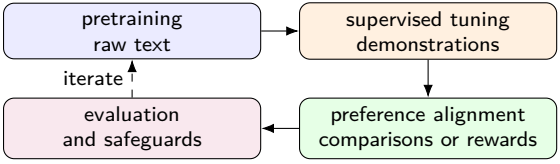
A plausible intermediate trace can still support a wrong answer and may not faithfully reveal the model's internal cause.

Source: Wei et al. (2022), chain-of-thought prompting; Promontory AI (2026), GWU practicum guest lecture.

Alignment shapes behavior after pretraining

- **Pretraining:** learn next-token prediction.
- **Supervised fine-tuning:** imitate curated demonstrations.
- **Preference alignment:** learn from comparisons or rewards through RLHF, DPO, or related objectives.
- **Evaluation and safeguards:** test intended behavior, misuse, and regressions.

Alignment changes behavior; it does not guarantee truth, safety, or universal agreement.



Alignment iterates through pretraining, supervised tuning, preference optimization, and evaluation.

Source: Ouyang et al. (2022), RLHF; Rafailov et al. (2023), DPO; Promontory AI (2026), GWU practicum guest lecture.

Putting the pieces together

- **Tokenization and encoding** turn text into token IDs—numbers a model can process.
- **Embeddings** map token IDs to learned vectors that represent useful features and similarities. Positional information represents token order.
- **Transformer blocks** combine self-attention, position-wise neural networks, residual connections, and normalization to update those vectors.
  - Feed-forward networks process each token position separately.
  - Self-attention mixes information across permitted positions; a causal language model uses the current and earlier tokens.
  - Stacked blocks learn increasingly complex contextual patterns. Many LLMs use decoder-only stacks.
- **Prediction** maps the final hidden vector to one score per vocabulary token, then converts the scores into probabilities.
- **Generation** selects or samples a next token, appends it to the sequence, and repeats the process.
- **Inference-time scaling and reasoning** can spend more computation on longer reasoning, multiple candidates, search, or verification before returning a response.
- **Alignment** steers next-token behavior toward human instructions, preferences, and safety goals, but does not guarantee truth or safety.

References

Holtzman, Ari, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi (2020). "The Curious Case of Neural Text Degeneration". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rygGQyrFvH.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv: 1301.3781. URL: https://arxiv.org/abs/1301.3781.

Ouyang, Long, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. (2022). *Training Language Models to Follow Instructions with Human Feedback*. arXiv: 2203.02155. URL: https://arxiv.org/abs/2203.02155.

Promontory AI (2026). *Engineering the Modern AI Stack: GWU Practicum Guest Lecture*. URL: https://promontoryai.com/gwu-practicum-26/.

Rafailov, Rafael, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn (2023). *Direct Preference Optimization: Your Language Model Is Secretly a Reward Model*. arXiv: 2305.18290. URL: https://arxiv.org/abs/2305.18290.

References

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 1715–1725. URL: https://aclanthology.org/P16-1162/.

Snell, Charlie, Jaehoon Lee, Kelvin Xu, and Aviral Kumar (2024). *Scaling LLM Test-Time Compute Optimally Can Be More Effective Than Scaling Model Parameters*. arXiv: 2408.03314. URL: https://arxiv.org/abs/2408.03314.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou (2022). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems* 35, pp. 24824–24837. arXiv: 2201.11903. URL: https://arxiv.org/abs/2201.11903.